```
65        // the following statement was used for debugging
66        // console.log( strReturn ) ;
67        return strReturn ;
68      }
69
70      // This function displays the table in whatever order it is currently sorted.
71      var DisplayTable = function( nSortColumn ) {
72        // Loop through the entire array, displaying the subarray in each top-level element
73        // in a separate row.
74        for ( var k = 0 ; k < arrCSSD.length ; k++ ) {
75          document.writeln( "<tr>" ) ;
76          document.writeln( "  <td>" + arrCSSD[k][0] + "</td>" ) ;
77          document.writeln( "  <td>" + arrCSSD[k][1] + "</td>" ) ;
78          document.writeln( "  <td>" + arrCSSD[k][2] + "</td>" ) ;
79          document.writeln( "  <td>" + TimeSlot( arrCSSD[k][3] ) + "</td>" ) ;
80          document.writeln( "</tr>" ) ;
81        }
82
83        // Highlight the header cell of the column on which the data is sorted.
84        // Be careful!  The getElementsByTagName function returns a *collection*, which is
85        // similar to, but not quite the same as an *array*!
86        var colHeaderCells = document.getElementsByTagName("th") ;
87        colHeaderCells[nSortColumn-1].setAttribute(
88          "style", "background-color: lightgreen ; color: black" ) ;
89      }
90
91
92      //// FUNCTIONS TO SORT THE TABLE DATA ON THE DESIRED COLUMN
93
94      // Note:  The following 4 functions could be combined into a single function with some
95      // additional programming, but I have left them as separate functions for clarity.  We
96      // could also write additional functions which sort the specified columns in descending
97      // order.
98
99      // This function compares the first (0th) items in the subarrays, which are the
100     // ROOM NUMBERS.  It returns -1 if the 1st item should come before the 2nd. +1 if the
101     // 1st item should come after the 2nd, 0 if their current order is OK.
102     var fnRoomAscending = function( a, b ) {
103       if ( a[0] < b[0] ) {
104         return -1 ;
105       } else if ( a[0] > b[0] ) {
106         return +1 ;
107       } else {
108         return 0 ;
109       }
110     }
111
112     // This function compares the second (index 1) items in the subarrays, which are the
113     // INSTRUCTOR NAMES.  It returns -1 if the 1st item should come before the 2nd. +1 if
114     // the 1st item should come after the 2nd, 0 if their current order is OK.
115     var fnNameAscending = function( a, b ) {
116       if ( a[1] < b[1] ) {
117         return -1 ;
118       } else if ( a[1] > b[1] ) {
119         return +1 ;
120       } else {
121         return 0 ;
122       }
123     }
124
```

6

```
125       // This function compares the third (index 2) items in the subarrays, which are the
126       // COURSE NAMES.  It returns -1 if the 1st item should come before the 2nd. +1 if the
127       // 1st item should come after the 2nd, 0 if their current order is OK.
128       var fnCourseAscending = function( a, b ) {
129         if ( a[2] < b[2] ) {
130           return -1 ;
131         } else if ( a[2] > b[2] ) {
132           return +1 ;
133         } else {
134           return 0 ;
135         }
136       }
137
138       // This function compares the fourth (index 3) items in the subarrays, which are the
139       // TIME SLOTS.  Remember that the time slots are encoded as "A", "B", "C", "D", or "E".
140       // Just like the other functions, this one also returns -1 if the 1st item should come
141       // before the 2nd. +1 if the 1st item should come after the 2nd, 0 if their current
142       // order is OK.
143       var fnTimeAscending = function( a, b ) {
144         if ( a[3] < b[3] ) {
145           return -1 ;
146         } else if ( a[3] > b[3] ) {
147           return +1 ;
148         } else {
149           return 0 ;
150         }
151       }
152
153
154       //// FUNCTION TO RELOAD THE PAGE WITH A SPECIFIED SORT COLUMN
155
156       // This function is executed when the user clicks on a column header cell.  The
157       // parameter passed to this function is the number of the column on which we wish to
158       // sort the table.
159       var ReloadPage = function( nSortColumn ) {
160         // The following statement returns:
161         //    "/StudentResources/CodeSamples/ArrayExample_4.html"
162         var strThisPagePath = window.location.pathname ;
163
164         // To this we want to add a query parameter, which is part of a query string.
165         // The question mark begins the query string part of a URL.
166         // Each parameter is then written in name=value format.
167         // Thus, what we want is the page path returned by the above statement, then a
168         //   question mark, then the query parameter name ("SortColumn") followed by an
169         //   equals sign (=) and finally followed by the parameter value, which in our
170         //   current case is the number of the desired sort column passed to this function.
171         var strFullPagePath = strThisPagePath + "?SortColumn=" + nSortColumn ;
172
173         // The final step is to replace the current page with the full path that we just
174         // constructed.  In our case, this is the same page that we are currently viewing,
175         // but with a parameter specifying the desired sort column.
176         window.location.replace( strFullPagePath ) ;
177       }
178
```